WP3: GPS and Time-Tagging
Case Western Reserve University (Cleveland, USA)

Corbin Covault, Robert Halliday
Robert Sobin, Andrew Ferguson

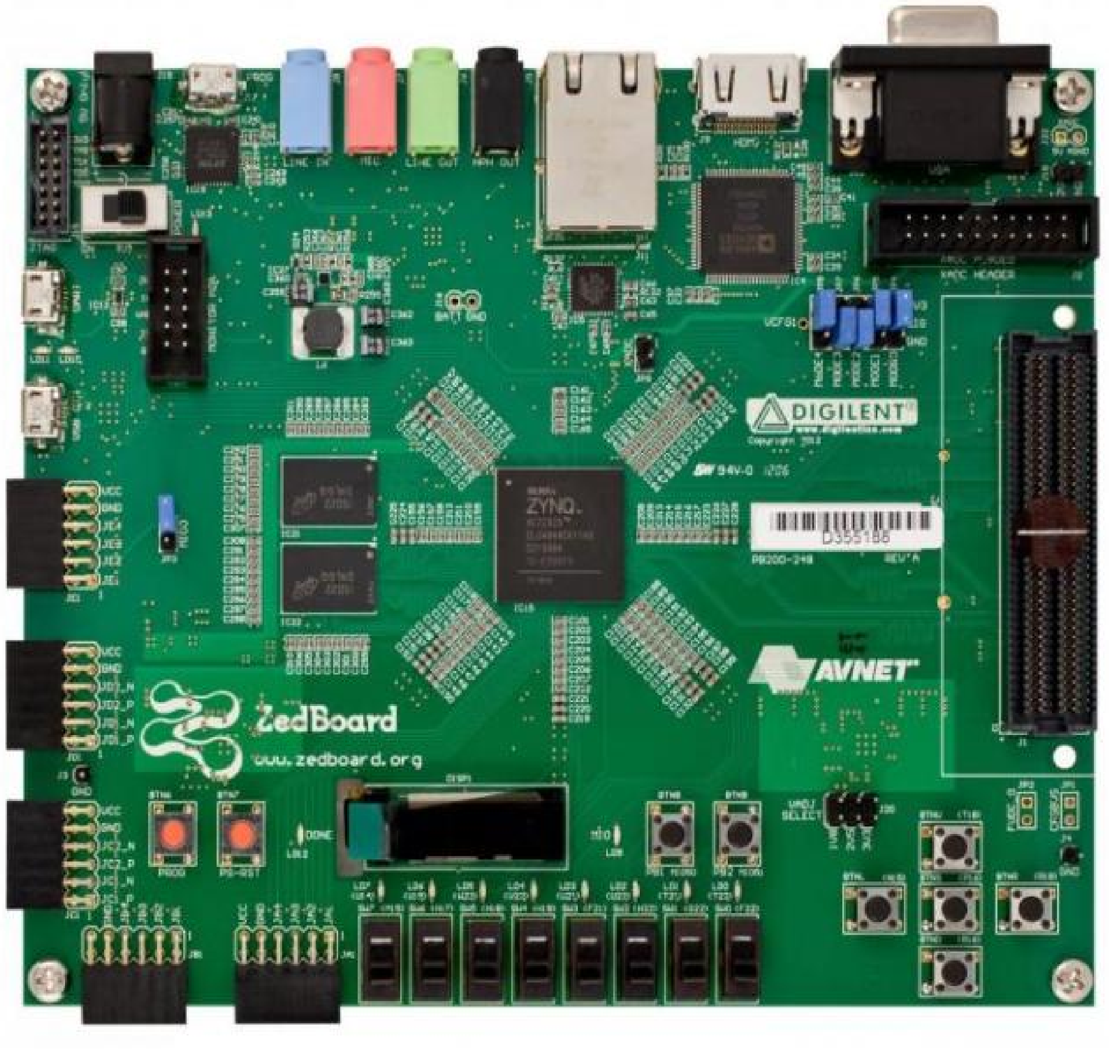
SDE Electronics CDR
Orsay, February 2015

WP3: GPS and Time-Tagging Tasks:

- **Hardware:** Select GPS receiver for use in the UUB. Also development of a Time-Tagging test stand to test, calibrate and characterize performance of GPS receivers.
- **Firmware:** Develop Time-Tagging modules for the UUB
- **Firmware:** Develop serial I/O interface between UUB and GPS daughterboard.
- **Software:** Develop OS drivers for control and communications
- **Software:** Re-write gpscntl code for GPS control, initialization, etc.
- **System:** Develop local expertise in establishing working system within a new board: “board bring-up” (OS, etc.)
- **Other:** Investigate negative impact of position-dependent drifts in atmospheric jitter on timing resolution within the array.

Work at CWRU on WP3 "Zedboard" development kit with Xilinx FPGA board, Zynq-7000 processing core

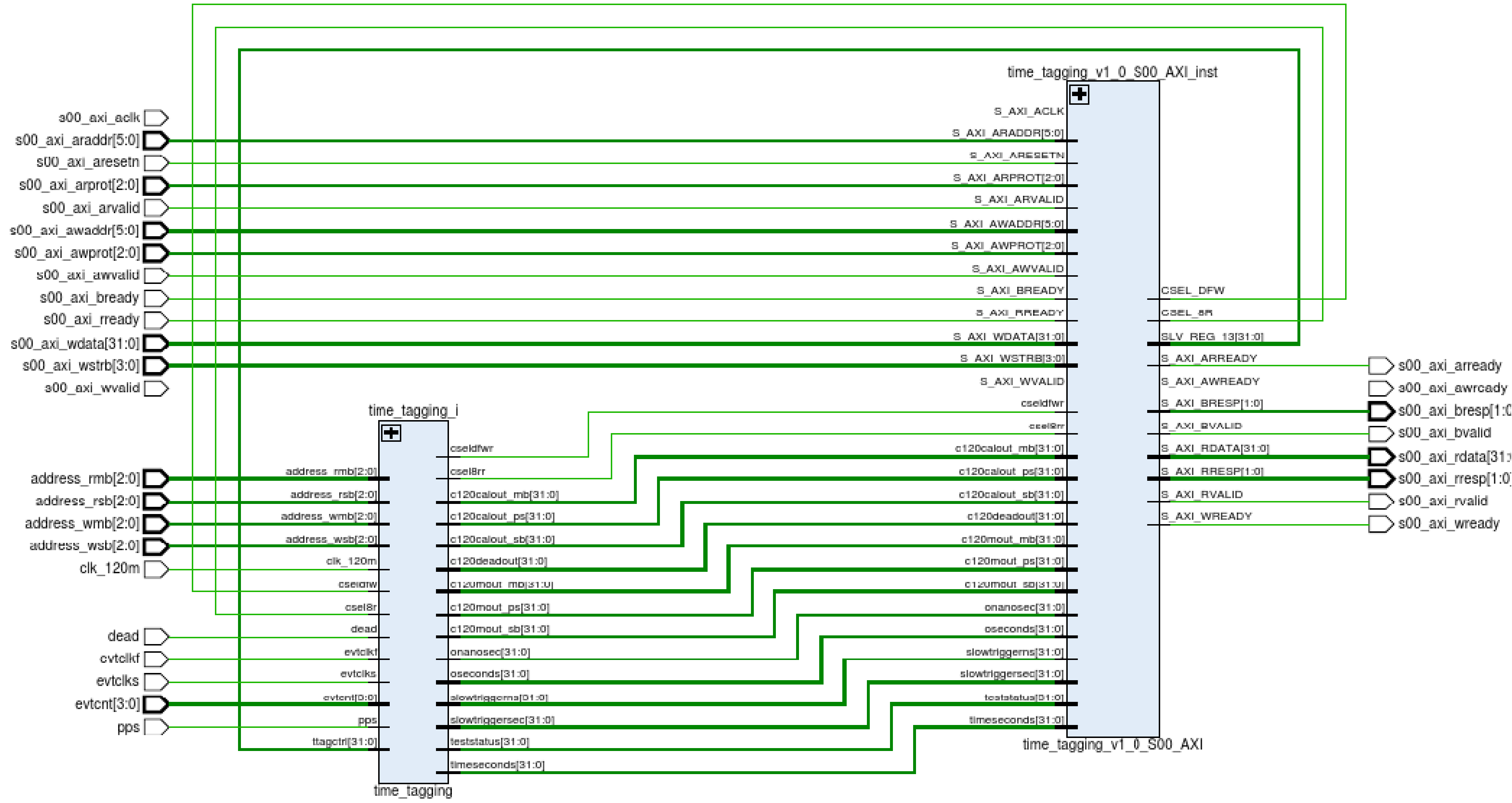
- **Vivado/SDK firmware development environment**
- **Firmware: Preliminary time-tagging module: DONE**
- **Firmware: serial I/O to GPS daughterboard: DONE**
- **Software: testbench time-tagging exercise program: DONE**
- **Software: firmware drivers and gpscntl for gps initialization: WORKING:**
- **Board bring up and OS installation: WORKING**



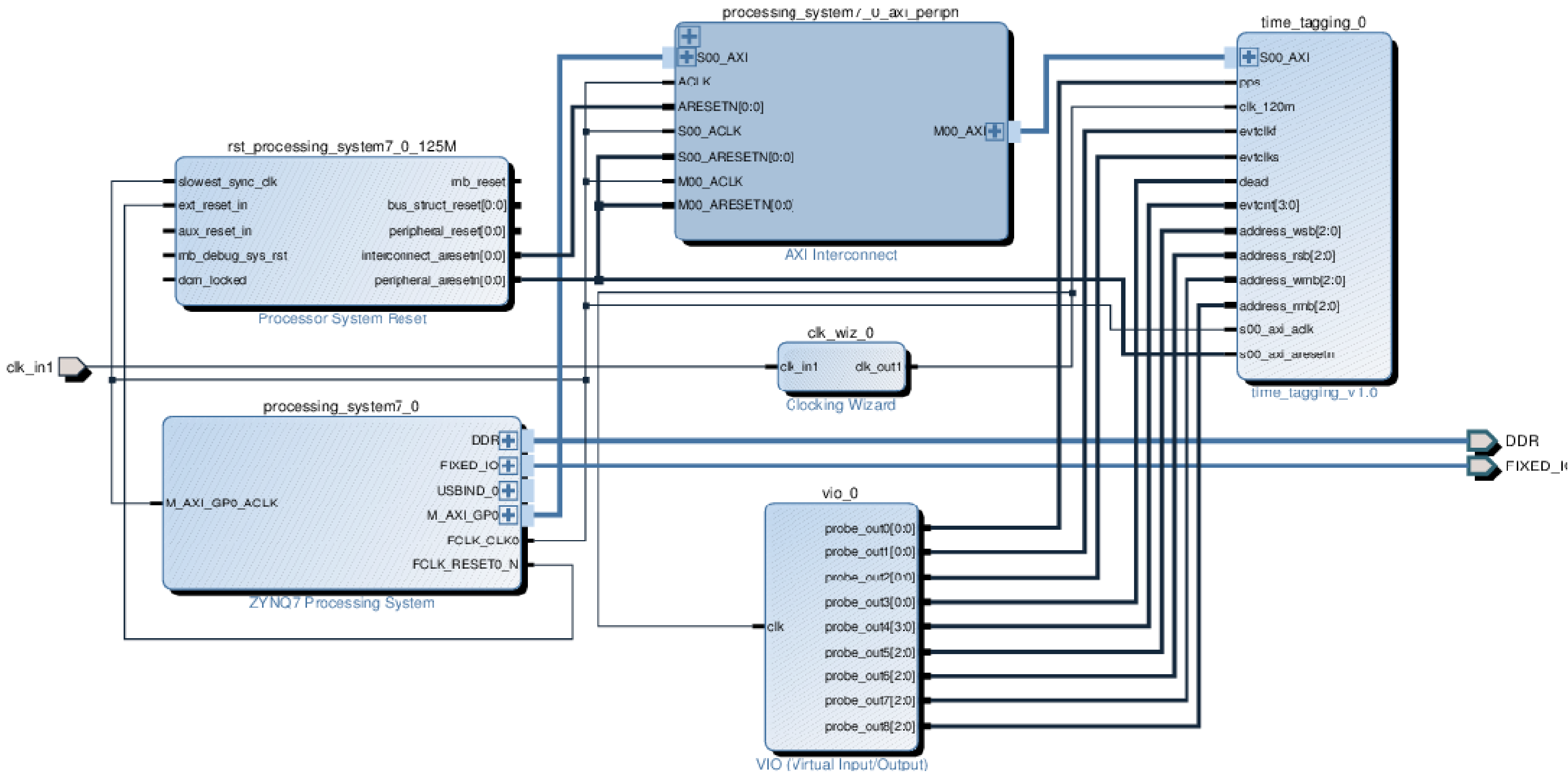
Time-tagging firmware module specifications:

- AXI slave peripheral interface: (16 registers available, 13 implemented)
- Preliminary register mapping scheme documented.
- Testing and modification of generic AXI peripheral interface handshaking: address cycle stretched to ensure reliable read given need to synchronize between clock domains.

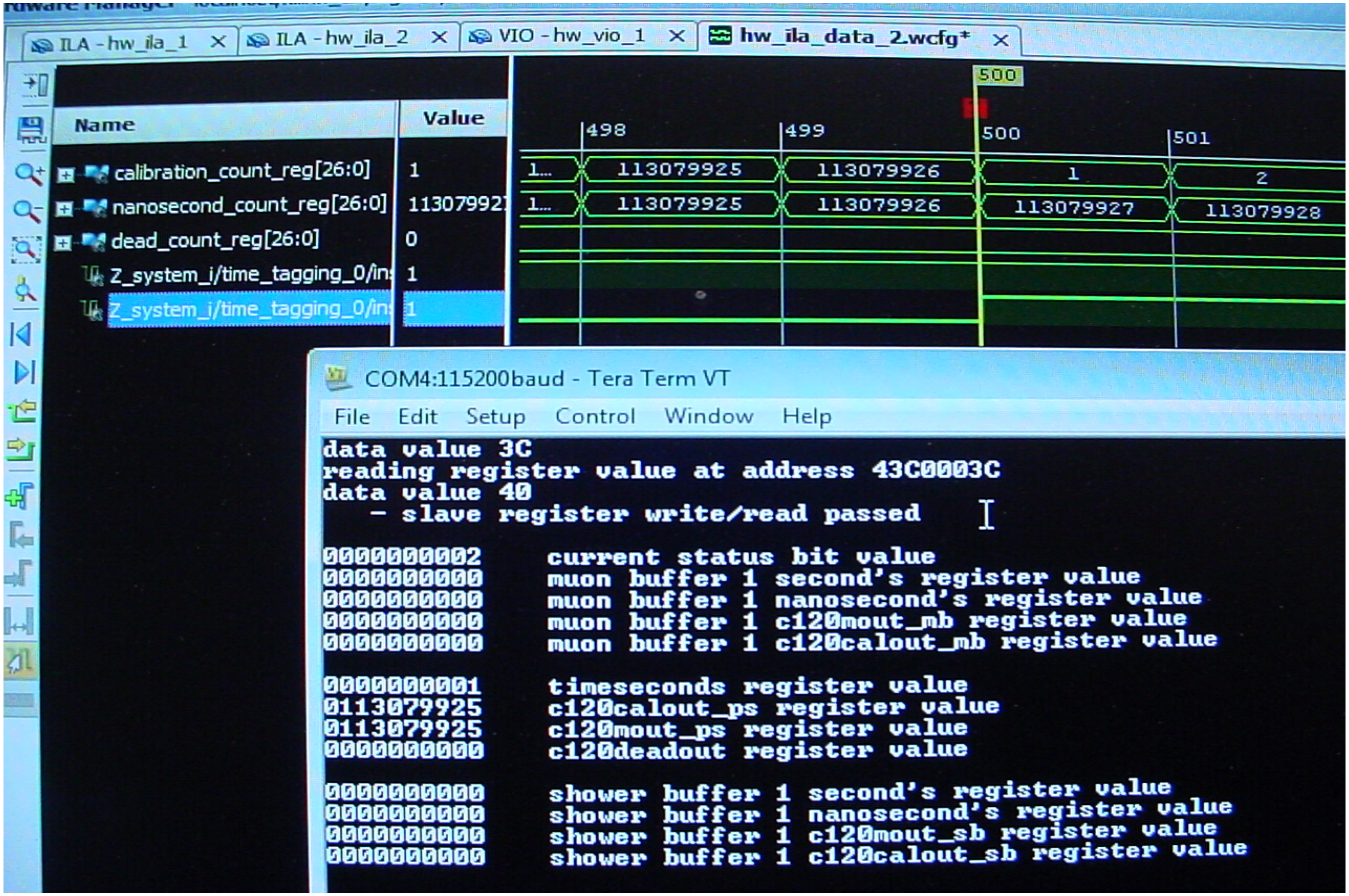
Schematics of time-tagging module with AXI interface



Block Diagram: Time-tagging module implementation example



Bench test using logic analyzer tool to verify register change of nanosecond counter upon 1 PPS from GPS



Name	Value	498	499	500	501
calibration_count_reg[26:0]	1	113079925	113079926	1	2
nanosecond_count_reg[26:0]	11307992	113079925	113079926	113079927	113079928
dead_count_reg[26:0]	0				
Z_system_i/time_tagging_0/in	1				
Z_system_i/time_tagging_0/in	1				

```
data value 3C
reading register value at address 43C0003C
data value 40
  - slave register write/read passed

0000000002     current status bit value
0000000000     muon buffer 1 second's register value
0000000000     muon buffer 1 nanosecond's register value
0000000000     muon buffer 1 c120mout_mb register value
0000000000     muon buffer 1 c120calout_mb register value

0000000001     timeseconds register value
0113079925     c120calout_ps register value
0113079925     c120mout_ps register value
0000000000     c120deadout register value

0000000000     shower buffer 1 second's register value
0000000000     shower buffer 1 nanosecond's register value
0000000000     shower buffer 1 c120mout_sb register value
0000000000     shower buffer 1 c120calout_sb register value
```

Status of Time-tagging module development:

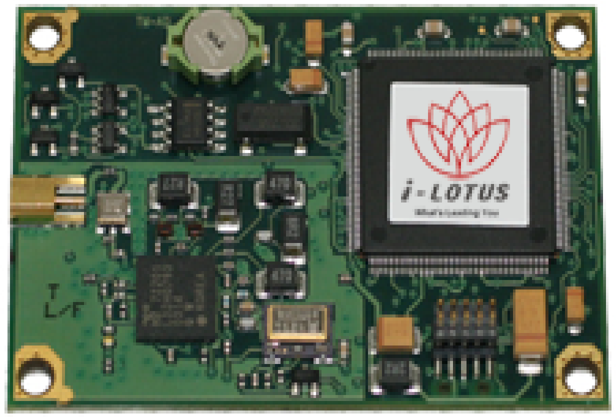
- DONE: Preliminary specification complete and documented.
- DONE: Preliminary modules implemented and tested.
- DONE: Slightly modified general AXI peripheral interface implemented.
- DONE: System tested and working on the bench using virtual I/O wrapper.
- WORKING: implement software access to module.

GPS Receiver Selection

Goal: Desired <5ns accuracy on 1 PPS, with good performance in varying temperatures.

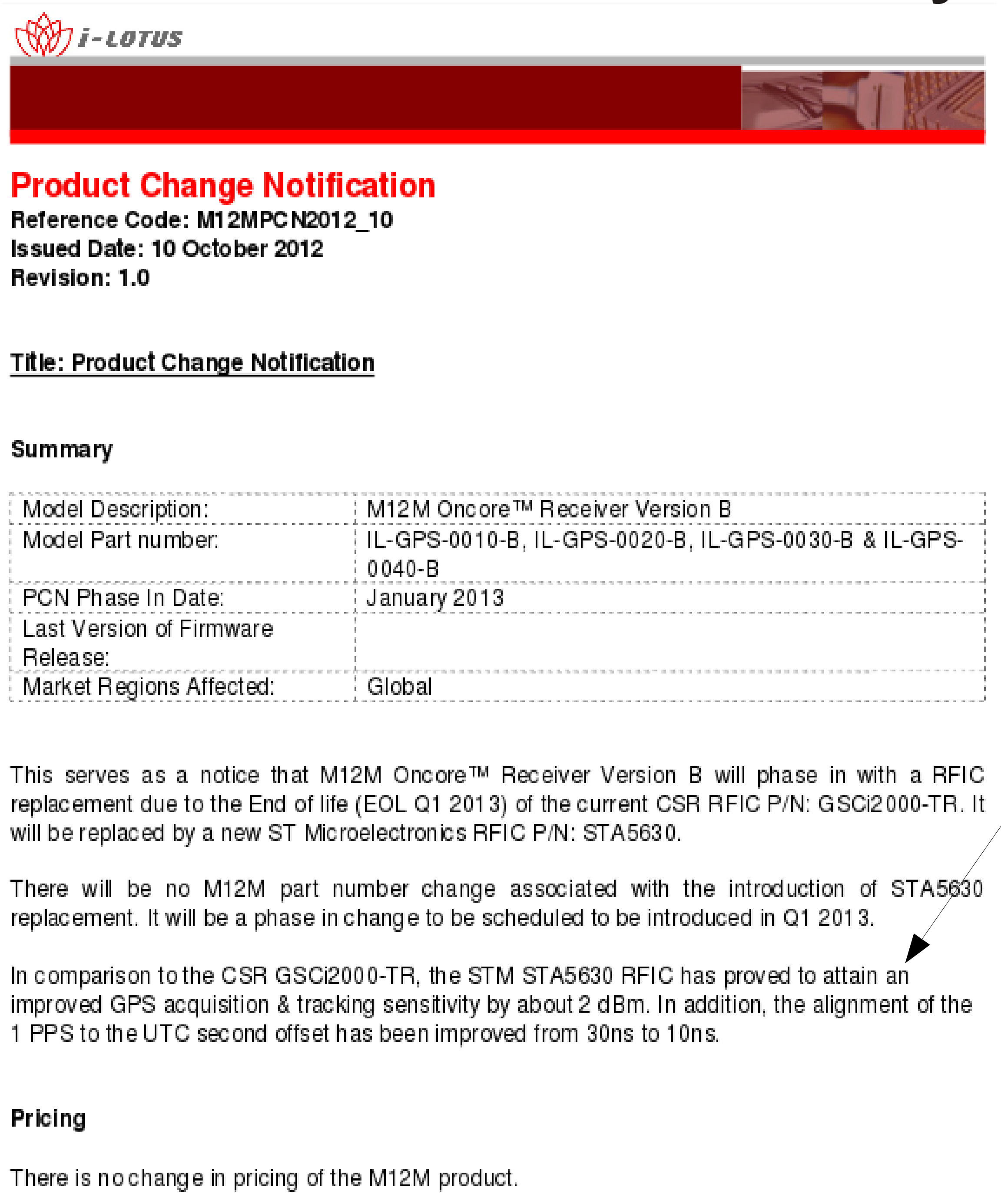
Backwards compatibility in the command set would be good.

- i-Lotus M12M was chosen with reported 2ns accuracy
- We also looked into an offering from Trimble in a similar price range
  - 2ns vs 15ns accuracy to 1-sigma, and Trimble does not use Motorola binary



CWRU has acquired and fully tested 20 of these receivers.

I-Lotus Product Change Notification for M12M, effective January 2013



i-LOTUS

Product Change Notification

Reference Code: M12MPCN2012_10
Issued Date: 10 October 2012
Revision: 1.0

Title: Product Change Notification

Summary

Model Description:	M12M Oncore™ Receiver Version B
Model Part number:	IL-GPS-0010-B, IL-GPS-0020-B, IL-GPS-0030-B & IL-GPS-0040-B
PCN Phase In Date:	January 2013
Last Version of Firmware Release:	
Market Regions Affected:	Global

This serves as a notice that M12M Oncore™ Receiver Version B will phase in with a RFIC replacement due to the End of life (EOL Q1 2013) of the current CSR RFIC P/N: GSCi2000-TR. It will be replaced by a new ST Microelectronics RFIC P/N: STA5630.

There will be no M12M part number change associated with the introduction of STA5630 replacement. It will be a phase in change to be scheduled to be introduced in Q1 2013.

In comparison to the CSR GSCi2000-TR, the STM STA5630 RFIC has proved to attain an improved GPS acquisition & tracking sensitivity by about 2 dBm. In addition, the alignment of the 1 PPS to the UTC second offset has been improved from 30ns to 10ns.

Pricing

There is no change in pricing of the M12M product.

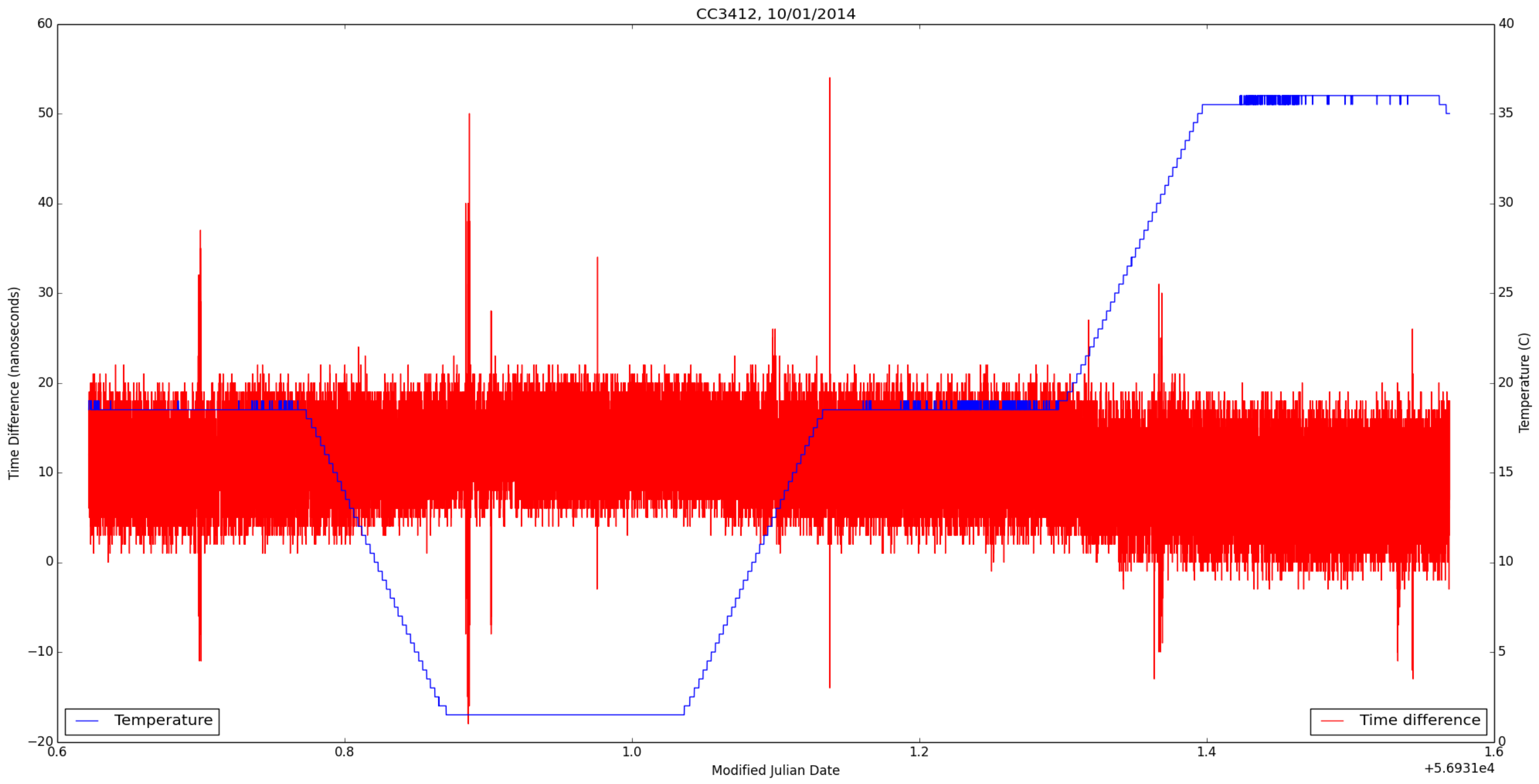
PCN suggests that M12M receivers manufactured after January 2013 might have improved accuracy, less drift relative to UTC.

Some M12M users also report improved temperature stability.

CWRU purchased 20 "new version" M12M receivers by mid-2014

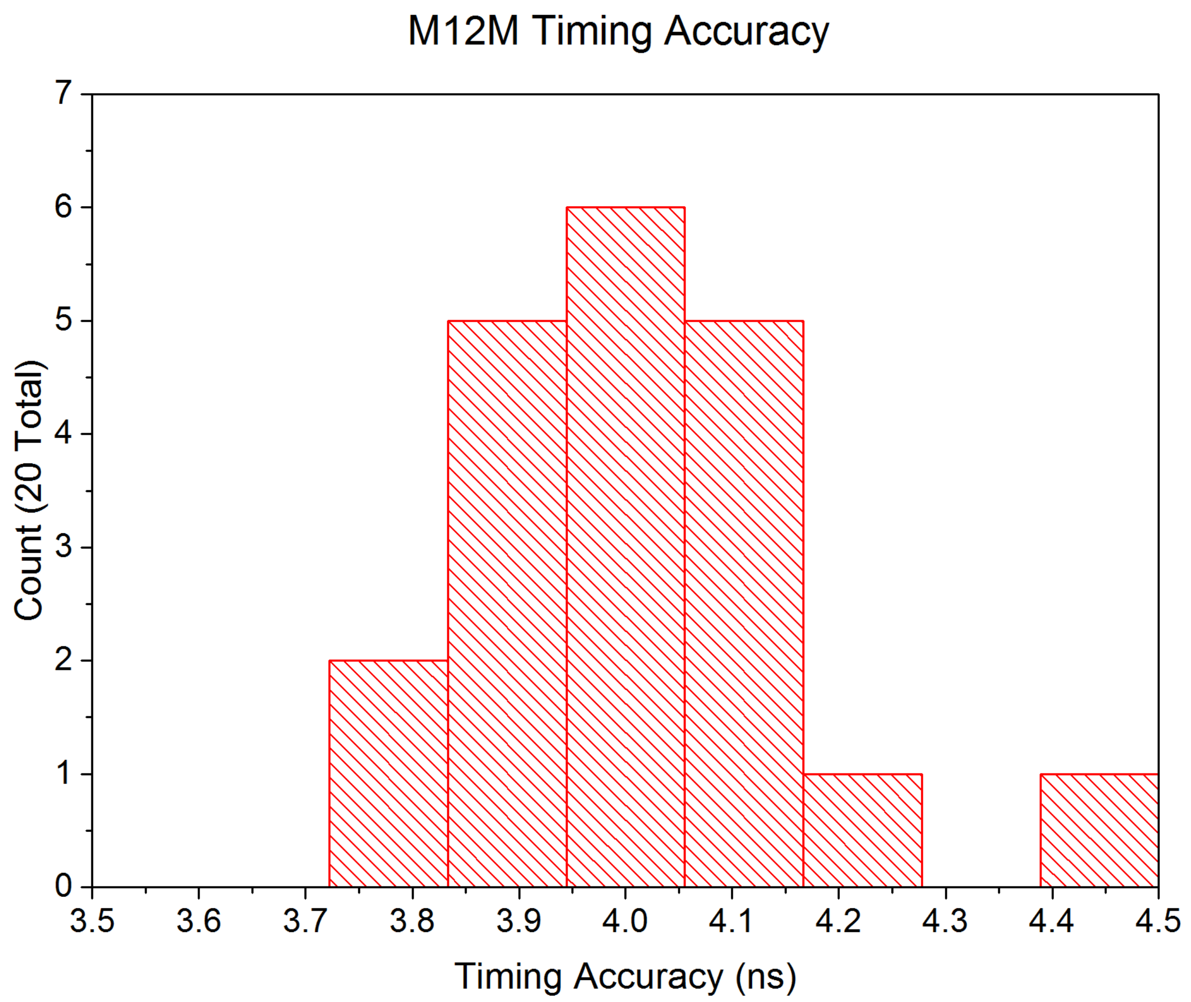
GPS Receivers, Temperature Response

Example: 25 hour temperature cycle test for M12M receiver



In early models we saw some issues with temperature stability. Prior models initially had 5-7ns PPS accuracy, but since Jan 2013 product change and after extensive testing we find ~4ns accuracy on a 250Mhz test stand.

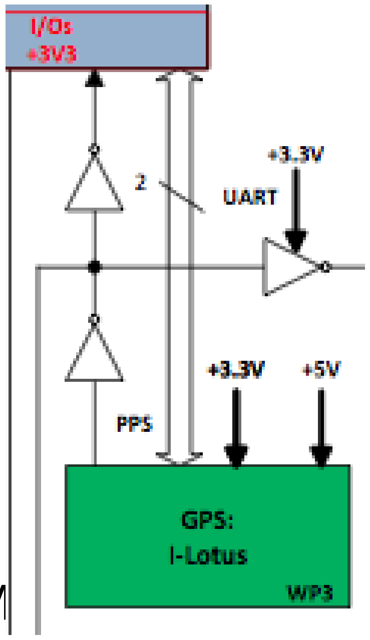
GPS Receivers, Timing Histogram

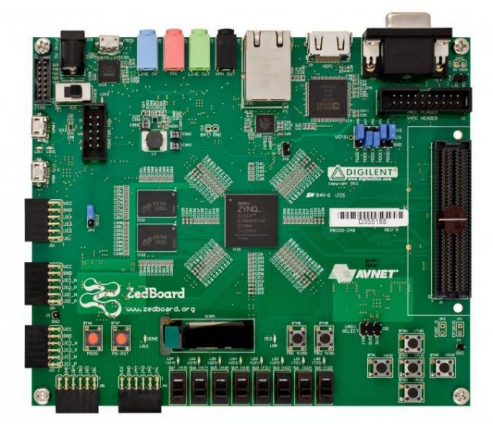


Note that these measurements are made on a test stand using a 150 Mhz counter. They were taken over a 25 hour test.

GPS Communications and Control

- Set-Up Serial Communications UART
  - Since the M12M goes directly into the Zynq, we will need a UART (Receiver/Transmitter) in the Programmable Logic to communicate with it
  - Will use model NS550(16550)
  - Completed and tested on ZedBoard (Zynq Evaluation Board)
- Rewrite GPS control module (Gpsctrl) for M12M
  - Controls start-up and messaging protocols, some of which have changed
  - Instead of 8 channel TRAIM, we now have 12 channel TRAIM requiring new data structures
  - Commands that need revision have been identified and rewriting is underway, soon to begin testing (2-4 weeks)





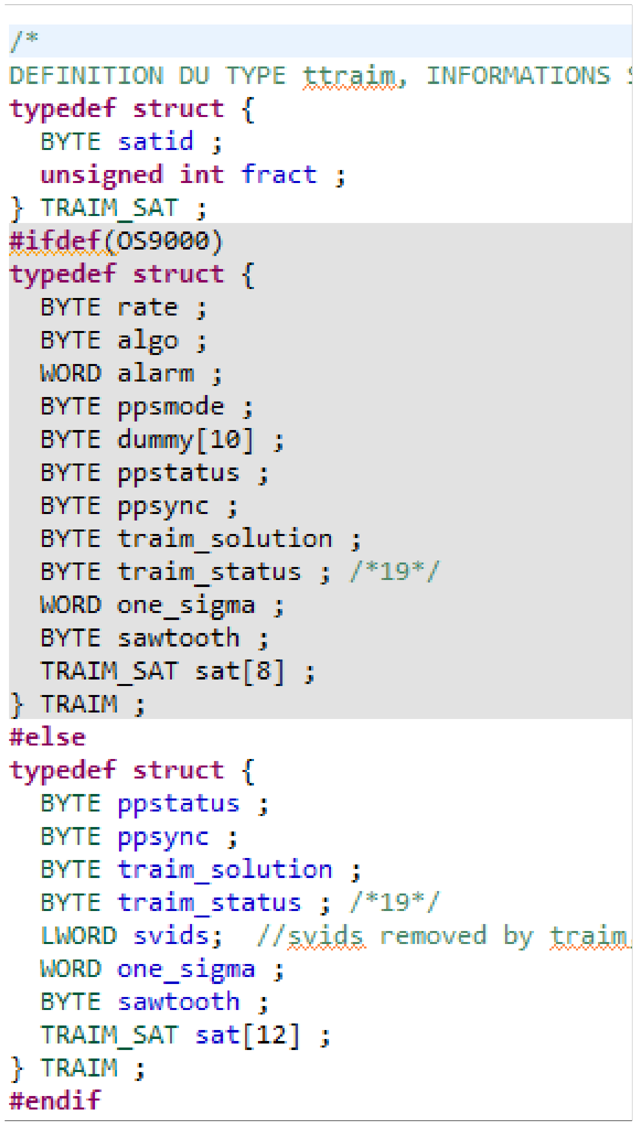
Serial UART

- Implementation is relatively simple, just need UART NS550 IP
  - Significantly more versatile than the UARTlite
- It can be interacted with using a Standalone driver(test bench), or a Linux driver(UUB)
- This allows Gpsctrl to function as it had been. The BAUD, Stop bits, etc. are the same from the UT to the M12M
  - 4 Commands dealing with TRAIM, Location and Time had/have to be changed

GPS Control Module

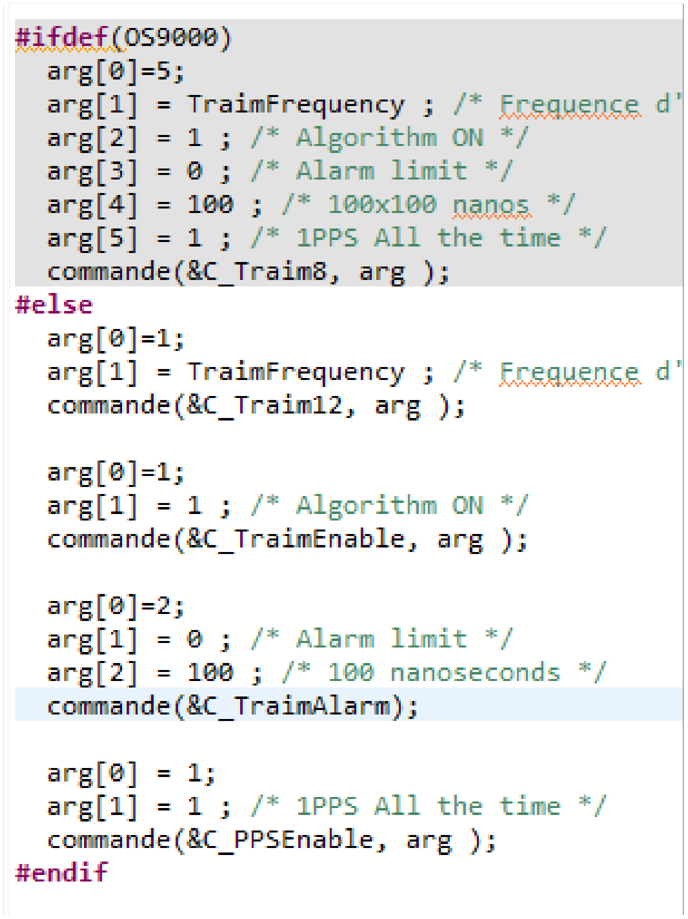
- Receiving messages:
  - Adapted TRAIM and Position data structures
  - Added IO handling for 12 channel commands (@@H*)
- Initialization:
  - TRAIM command splays out into many commands
  - Remapped position hold enable

TRAIM Data Structure:



```
/*
DEFINITION DU TYPE ttraim, INFORMATIONS S
typedef struct {
  BYTE satid ;
  unsigned int fract ;
} TRAIM_SAT ;
#ifdef(OS9000)
typedef struct {
  BYTE rate ;
  BYTE algo ;
  WORD alarm ;
  BYTE ppsmode ;
  BYTE dummy[10] ;
  BYTE ppstatus ;
  BYTE ppsync ;
  BYTE traim_solution ;
  BYTE traim_status ; /*19*/
  WORD one_sigma ;
  BYTE sawtooth ;
  TRAIM_SAT sat[8] ;
} TRAIM ;
#else
typedef struct {
  BYTE ppstatus ;
  BYTE ppsync ;
  BYTE traim_solution ;
  BYTE traim_status ; /*19*/
  LWORD svids;  //svids removed by traim
  WORD one_sigma ;
  BYTE sawtooth ;
  TRAIM_SAT sat[12] ;
} TRAIM ;
#endif
```

TRAIM Initialization:



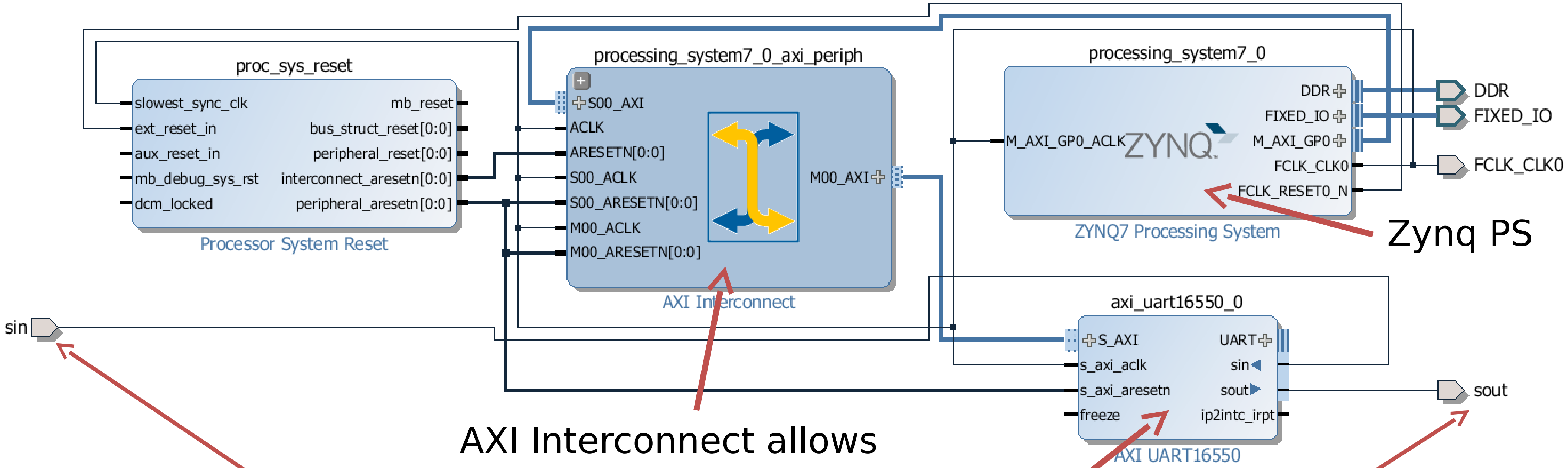
```
#ifdef(OS9000)
  arg[0]=5;
  arg[1] = TraimFrequency ; /* Frequence d'
  arg[2] = 1 ; /* Algorithm ON */
  arg[3] = 0 ; /* Alarm limit */
  arg[4] = 100 ; /* 100x100 nanos */
  arg[5] = 1 ; /* 1PPS All the time */
  commande(&C_Traim8, arg );
#else
  arg[0]=1;
  arg[1] = TraimFrequency ; /* Frequence d'
  commande(&C_Traim12, arg );

  arg[0]=1;
  arg[1] = 1 ; /* Algorithm ON */
  commande(&C_TraimEnable, arg );

  arg[0]=2;
  arg[1] = 0 ; /* Alarm limit */
  arg[2] = 100 ; /* 100 nanoseconds */
  commande(&C_TraimAlarm);

  arg[0] = 1;
  arg[1] = 1 ; /* 1PPS All the time */
  commande(&C_PPSEnable, arg );
#endif
```

UART Block Diagram



Zynq PS

AXI Interconnect allows many AXI devices on one Zynq Bus

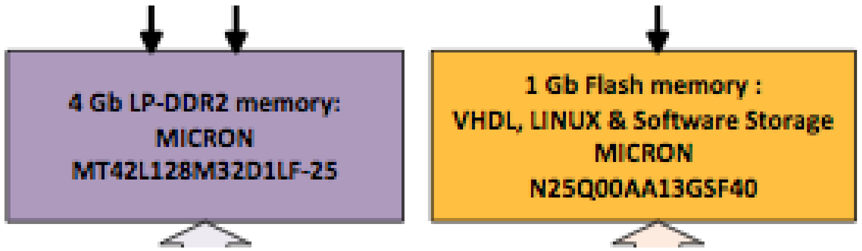
UART 16550

Serial In and Out made external for connection

Board Bring-Up

In Moving to the new board, we are choosing a new operating system. In order to run the operating system on the UUB, it must be preconfigured. Although we can't prepare the OS for the UUB yet, we can practice on the ZedBoard.

1. Hardware Image Bootstrapper (Vivado/SDK)
2. First Stage Bootloader (SDK)
3. Second Stage Bootloader (U-Boot)
4. Operating System (PetaLinux)



The hardware image is loaded to the board first, and then the board begins running through the stages of startup.

Hardware Image and First Stage Bootloader (FSBL)

- Once a Block Diagram and configuration are complete in Vivado, they can almost immediately be used to make a Board support package (BSP)
- Both of these are created in SDK
  - system_top.bit (From BSP)
  - zynq_fsbl.elf From (FSBL Template)
- The FSBL is a low level program that starts up the board and secures resources to launch the Second Stage Bootloader

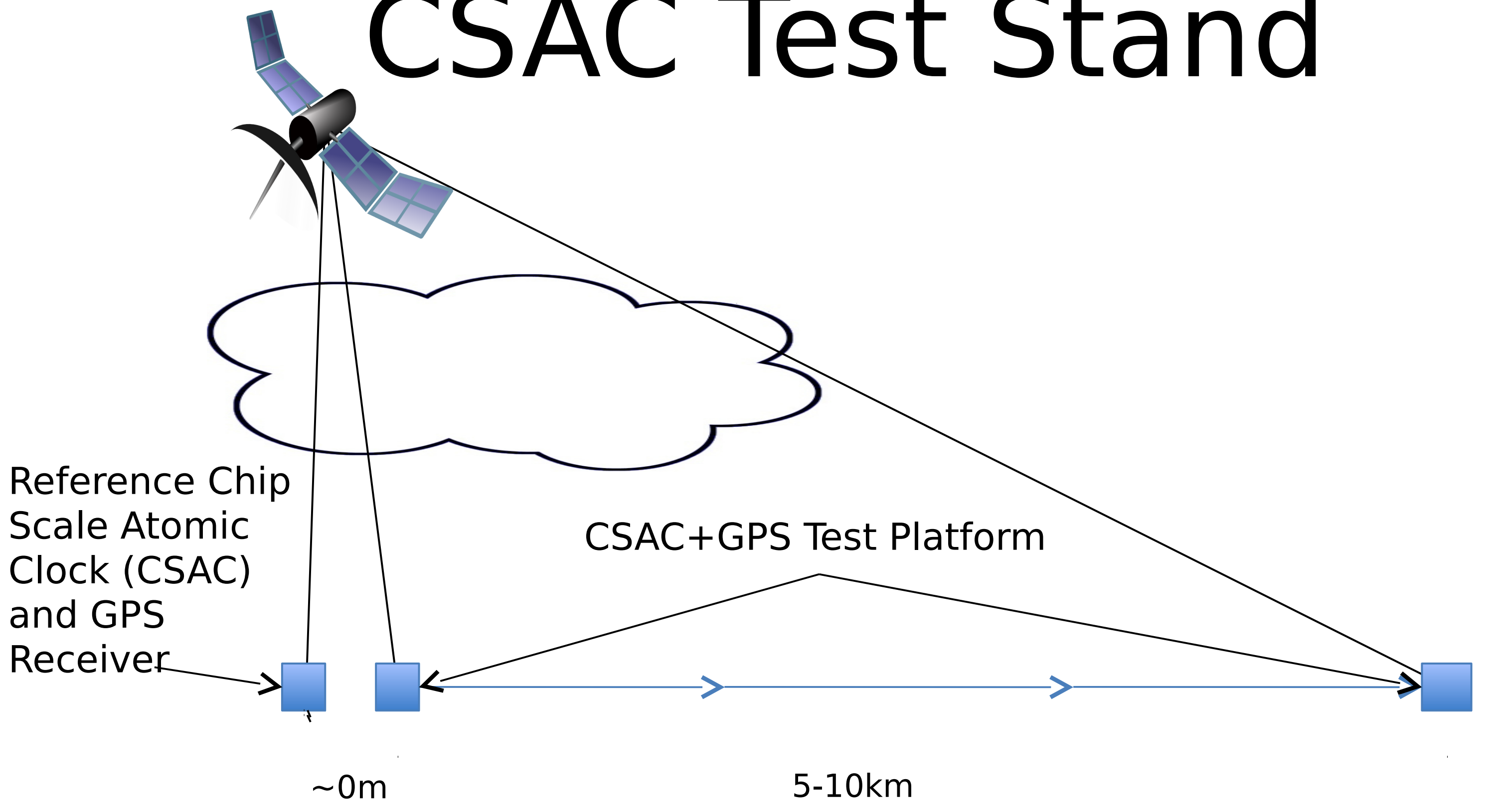
Second Stage Bootloader

- U-Boot is the most commonly used solution here. It is available for many systems and is most often cross-compiled
- Zynq is already supported for U-Boot; this already works on the ZedBoard
- The second stage bootloader prepares the system to launch Linux.
- To create this, we will add a makefile option to U-Boot for the UUB. Most of the standard options (zynq_common.h) will be fine.

Operating System

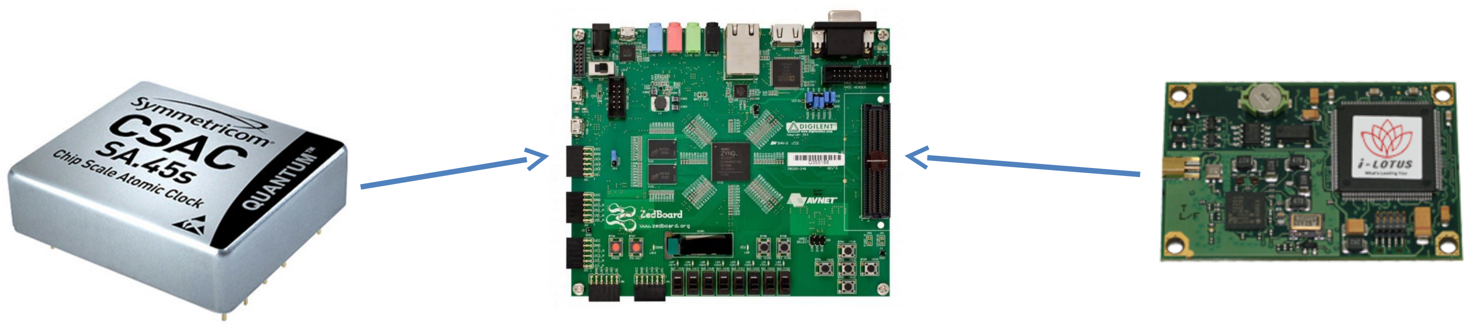
- The operating system requires a few elements, but our concern is generating a device tree.
- PetaLinux is the Xilinx supported option. It is light weight and comes with a Zynq BSP, along with its own version of SDK, which creates the device tree and boot image.
- All stages are then given priority during boot and tied together with SDK.

CSAC Test Stand



We would like to measure the relative drift of two GPS units as a function of the distance between them. This will give us a good idea of if/how this is affecting the array's timing. To do this we will use atomic clocks as an independent time reference and create a test platform which we can synchronize with a reference and then move to varying distances from the lab.

CSAC Test Stand



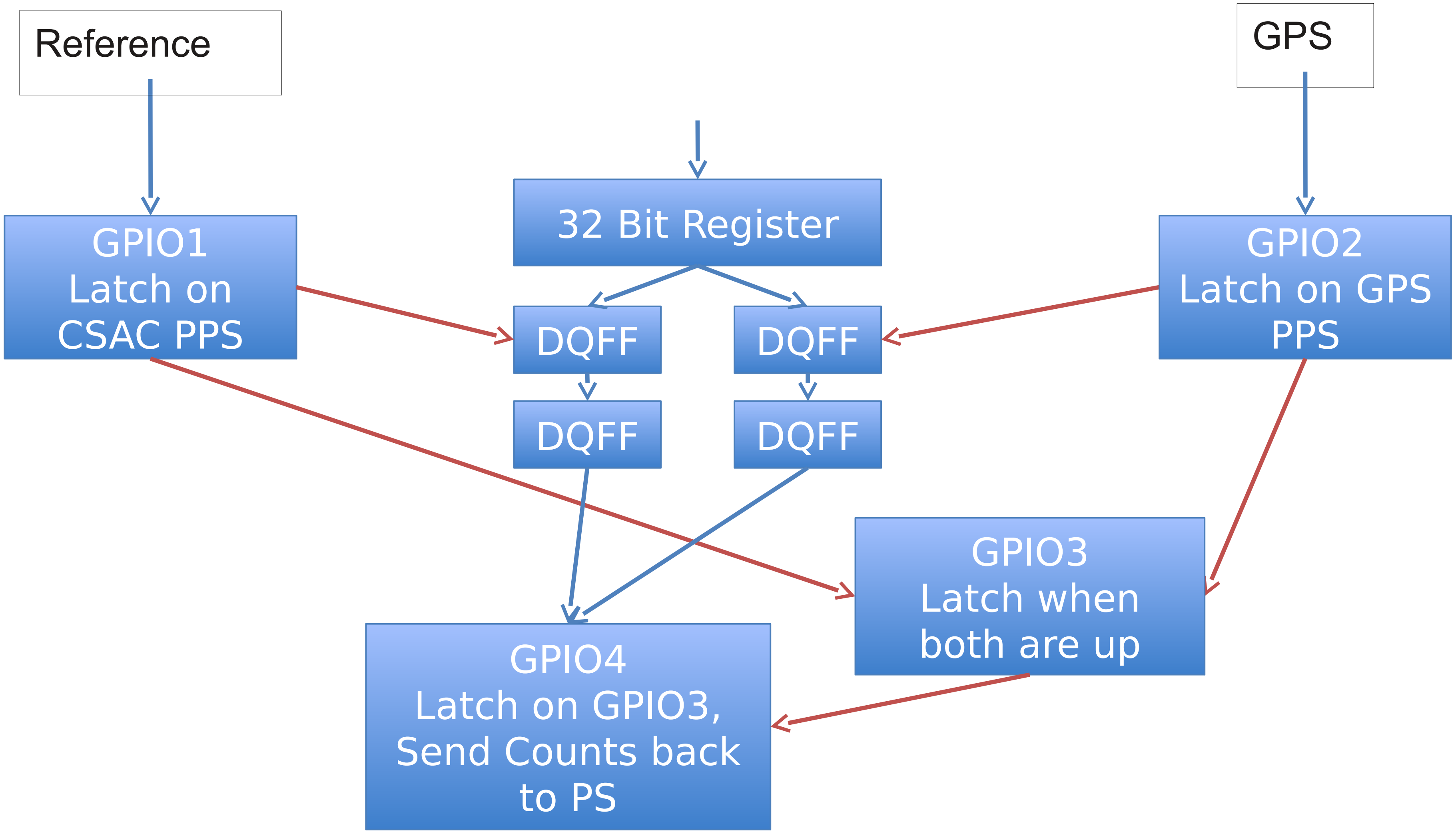
CSAC
78ps

ZedBoard
600Mhz, 1.8ns

M12M GPS
Timing
Receiver
2ns

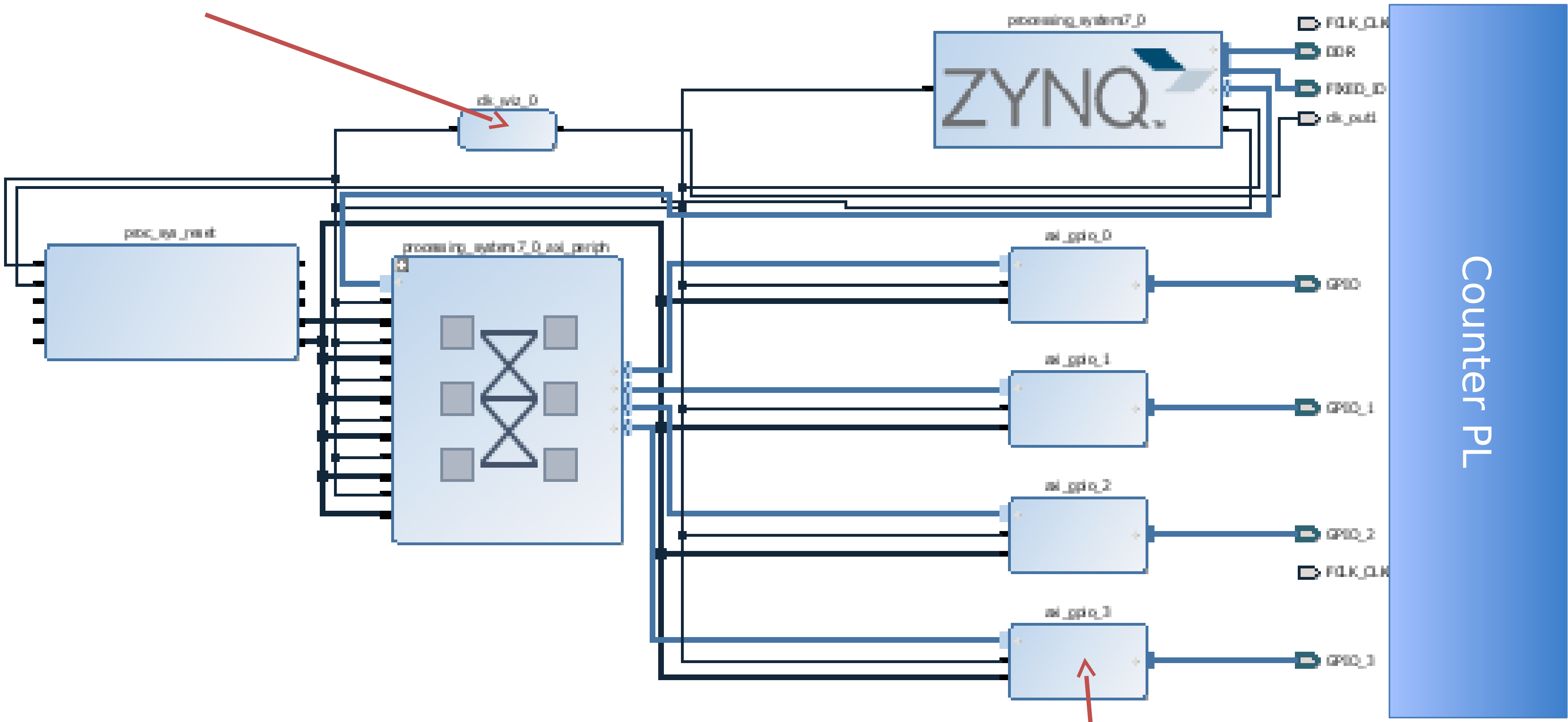
- This setup will also double as a more advanced test stand for the incoming GPS receivers which will be able to measure down to their stated maximum accuracy
- At this point we have a 600Mhz oscillator implemented on the ZedBoard (supposed max 800Mhz) and upon the arrival of the atomic clock(s) we will start assembling the test platform.

Counter PL for CSAC Test Stand



Zynq PL for CSAC Test Stand

Counter Clock



General Purpose Input/Output (GPIO)